Lecture 20

Plan:

1) Finish LCIS algo.
2) Matroid intersection polytope

Matroid intersection polytope

- Let $M_1 = (E, I_1)$, $M = (E, I_2)$ be two matroids.

- analogously to the matroid polytope, let

$$X = \{1_S \in \{0,1\}^E : S \in I_1 \cap I_2\}.$$

ie. $X \subseteq \mathbb{R}^E$ set of indicator vectors of $I_1 \cap I_2$.

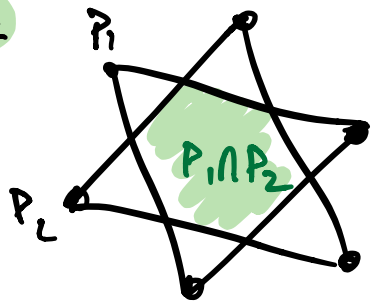
- Define the **matroid intersection polytope** $P_{M_1,M_2} := \text{conv}(X)$.

- Main result: P_{M_1,M_2} is the intersection $P_{M_1} \cap P_{M_2}$ of the matroid polytopes for M_1, M_2.

- This is surprising! For general polyhedra P_1, P_2,

$$\text{conv}(\text{vertices}(P_1) \cap \text{vertices}(P_2)) \neq P_1 \cap P_2 !$$

e.g.



P_1, P_2 share no vertices, but $P_1 \cap P_2 \neq \emptyset$!

- In terms of inequalities!
- Recall matroid polytope: if r rank function of M,

$$P_M = \{x \in \mathbb{R}^E : x(S) \leq r(S) \ \forall S \subseteq E, \ x_e \geq 0 \ \forall e \in E\}.$$

- $P_{M_1} \cap P_{M_2}$ has constraints for both; hence

Theorem: Let $P = P_{M_1} \cap P_{M_2}$, i.e.

$$P = \{x \in \mathbb{R}^E : x(S) \le r_1(S) \ \forall S \subseteq E$$
$$x(S) \le r_2(S) \ \forall S \subseteq E$$
$$x_e \ge 0 \quad \forall e \in E\}$$

Then

$$\boxed{P_{M_1, M_2} = P.}$$ i.e. $P_{M_1 M_2} = P_{M_1} \cap P_{M_2}$.

Proof: Plan:

- Like second proof for matroid polytope use vertex integrality.
- Integrality suffices by the usual logic:
  - clearly $\text{conv}(X) \subseteq P$, b/c $X \subseteq P_{M_1}, P_{M_2}$.

▷ On the other hand, if P integral then $P \subseteq \text{conv}(X)$ because integral points in P_{M_1}, P_{M_2} are indicators indep sets in M_1, M_2, resp.

- Again, A s.t. $P_{M_1 M_2} = \{x : Ax \leq b, x \geq 0\}$ is <u>not</u> totally unimodular.
- But the submatrices describing vertices <u>ARE</u> T.U.

Let x^* be an extreme point of P.

- We know x^* uniquely characterized by which inequalities tight.

- For $i \in \{1,2\}$, let

$$T_i = \{S \subseteq E : x^*(S) = r_i(S)\}$$

i.e. sets s.t. rank constraint tight in M_i,

- Let $J = \{e : x^*_e = 0\}$.

- Then x^* is unique solution to linear system

$$x(S) = r_1(S), \quad S \in T_1,$$

$$x(S) = r_2(S), \quad S \in T_2.$$

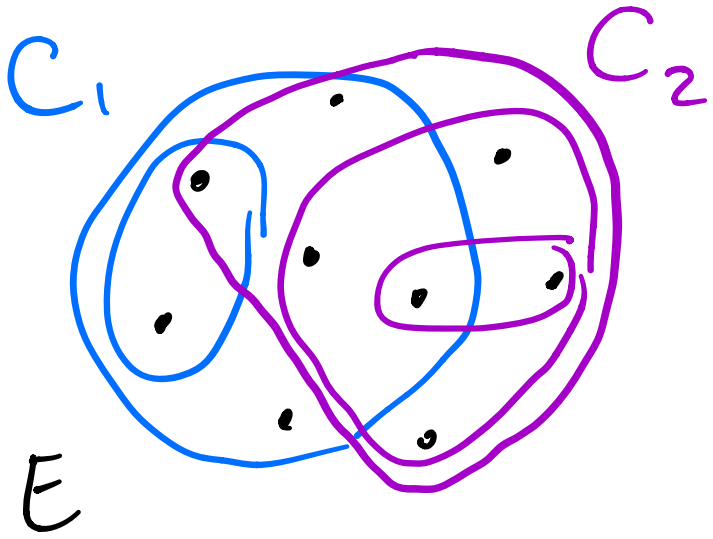
$$x_e = 0 \qquad \forall e \in J.$$

- this is the intersection of two faces F_1, F_2 of P_{M_1}, P_{M_2}, resp:

$$F_i = \{x \in P_{M_i} : x(S) = r_i(S) \ \forall S \in T_i\}.$$

- But recall from lec 17 that T_i can be replaced by a chain C_i w/out changing F_i!

e.g.



- Thus, assume T_i is chain C_{i_j}

x^* solution to

$$x(S) = r_1(S), \quad S \in C_1,$$

$$x(S) = r_2(S), \quad S \in C_2.$$

$$x_e = 0 \qquad \forall e \in J.$$

- This is $Ax = b$ for $b \in \mathbb{Z}$,

Claim: A T.U. $\Rightarrow x^*$ integral.

- Why is A T.U.? Rows of A are 1s for S in C_1 or C_2.

e.g.

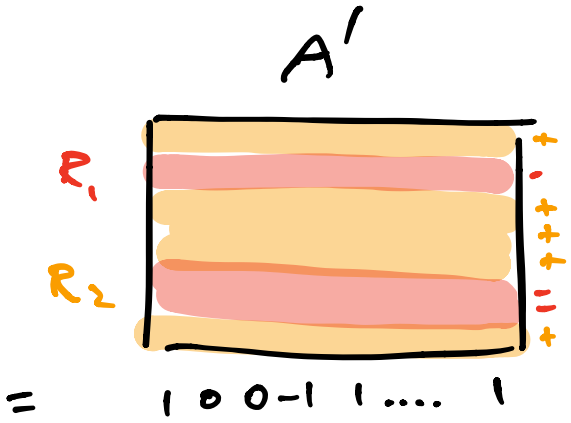
$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{matrix} \} C_1 \\ \\ \\ \} C_2 \\ \\ \\ \end{matrix}$$

- We use discrepancy to prove.

- Recall: A T.U. $\Leftrightarrow \forall$ submatrices A' of A, $\exists$ partition R_1, R_2 of rows s.t.

$\sum_{i \in R_1} a_i - \sum_{i \in R_2} a_i$ has entries in $\{-1, 0, +1\}$.

e.g.



- Consider submatrix A' of A; corresponds to subchains C_1', C_2', (hence has same form).

- Assign R_1, R_2 in following way.

  - Assign largest set of C_1' to R_1, then alternately assign remain elts of C_1 to R_2, R_1.

e.g. C_1'

sum will have entries in $\{0, 1\}$.

▷ For C_2', assign the opposite way:

e.g. C_2'

Sum has entries in $\{0, -1\}$.

- Overall, sum has entries in $\{-1, 0, 1\}$! □.

Matroid intersection optimization

- Given a cost function $c : E \to \mathbb{R}$, can we efficiently compute

$$\max_{S \in I_1 \cap I_2} c(S) \; ?$$

- For just one matroid, greedy alg.
- For $c = 1$, just LCIS.
- For perfect matchings, Hungarian alg.
- Can also compute min cost LCIS.

Exercise: equiv to max cost indep. set for $c' = K - c$ for some large K.

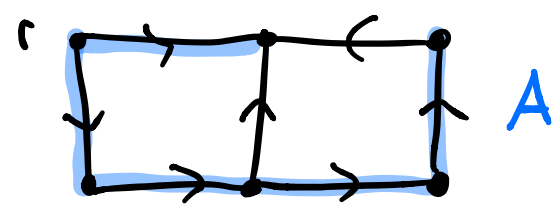
- In general, YES!
  - ellipsoid algorithm
  - Complicated primal-dual algos.
    → Strongly poly time.

- **Today:** simpler primal-dual algo for minimum cost arborescence.

Min-cost arborescence

- **Recall:** given directed graph D & vertex r, **arborescence** A is spanning tree directed away from r.

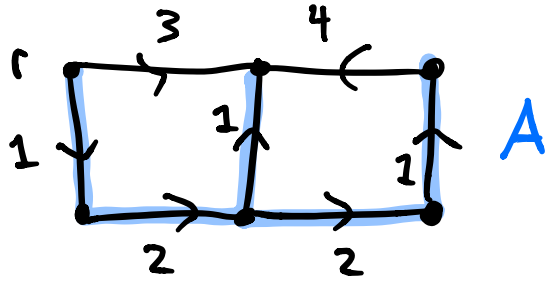
e.g.



- min-cost arborescence:

$$\min_{A \text{ arborescence}} \sum_{e \in A} c(e)$$

e.g.



- e.g. edges = roads to fix
  r = distribution center,
  Cost = expense of fixing roads

- First, **I.P. formulation:**

assume c nonnegative.

$$OPT = \min_{x \in \mathbb{R}^E} \sum_{e \in E} c_e x_e$$

subject to $\sum_{e \in \delta^-(S)} x_e \geq 1 \quad \forall S \subseteq V - r$

cut (no cuts)

$\sum_{e \in \delta^-(v)} x_e = 1 \quad \forall v \in V - r$

indegree (every vertex except r has 1 incoming edge)

$x_e \in \{0, 1\}$

- Check: only solutions are 1_A, $A \subseteq E$ arborescence.

- Miraculously, we'll show even w/out integrality constraint or indegree constraint, optimal solutions still arborescences.

- I.e. the following L.P. has an optimizer 1_A, A arborescence.

$$LP = \min_{x \in \mathbb{R}^E} \sum_{e \in E} c(e) x_e$$

subject to

$$\sum_{e \in \delta^-(S)} x_e \geq 1 \quad \forall S \subseteq V - r$$

(primal)

$$x_e \geq 0 \quad \forall e \in E$$

(note $OPT \leq LP$; fewer constraints).

- Dual LP is

$$LP = \max \sum_{S \subseteq V-r} y_S$$

subject to

$$\sum_{S : e \in \delta(S)} y_S \leq c(e) \quad \forall e \in E$$

(dual)

$$y_S \geq 0 \quad \forall S \subseteq V-r$$

- **Algorithm sketch**: construct
  - ▷ arborescence A,
  - ▷ dual feas y

satisfying complementary slackness.

Then $c(A) = LP$, but $OPT \leq LP$,

$\Rightarrow c(A) = OPT.$

- Complementary slackness for 1_A, y says:

  a.) $y_S > 0 \Rightarrow |A \cap \delta^-(S)| = 1$,

  b.) $e \in A \Rightarrow \sum_{S : e \in \delta^-(S)} y_S = c(e)$.

- Two phases of algorithm:

  1) Construct

  ▷ dual feas. y

  ▷ set F of edges containing directed path from r to every root.

  (maybe not spanning tree).

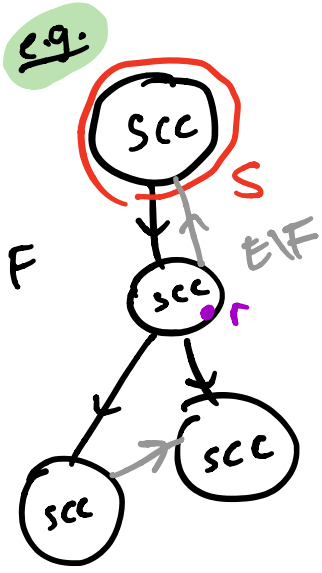
  & such that y, F satisfy (b).

2) Remove unnecessary edges from F, get arborescence satisfying (a) also.

Phase 1 Initialize $F = \emptyset$, $y = 0$, counter $R = 1$.

▷ While not everything reachable using F:

▷ select S s.t.



i) F strongly connected in S (every vertex can reach every other)

ii) $F \cap \delta^-(S) = \emptyset$.

S is a "source" in strongly connected component decomposition of F.

▷ increase y_S until new inequality

$$\sum_{S : e_k \in \delta^-(S)} y_S \leq c(e_k)$$

becomes equality; note $e_k \notin F$.

(y remains dual feas, satisfies (b) with F b/c it did before).

▷ $F \leftarrow F + e_k$, $k \leftarrow k+1$.

new F, y doesn't violate (b) because e_k tight.

▷ Return F, y satisfying (b).

Phase 2: eliminate as many edges as we can in reverse order they were added:

- For $i = 1 \dots k$:
  - If $F - e_i$ contains directed path from r to every vertex, $F \leftarrow F - e_i$.
- Return $A = F$.

Claim 1: A arborescence.

Pf:
- We'll show $|A| = |V| - 1$ & $d^-(v) = 1$ $\forall v \in V - r$.
- If indegree < 1, contradicts reachability in A. or $|A| < |V| - 1$
- if $|A| > |V| - 1$, would be collision:  e_i.

- Suppose $i < j$. In reverse delete, would have removed e_j b/c any vertex reachable through e_j was reachable through e_i also. □

finally:

Claim 2: Condition (a) of complementary slackness holds:

a.) $y_S > 0 \Rightarrow |A \cap \delta^-(S)| = 1$.

Pf: Assume not; $\exists S$ s.t.

$$y_S > 0 \quad \& \quad |A \cap \delta^-(S)| > 1.$$

- S was chosen by algorithm at some step k of Phase 1; we added e_k to F.

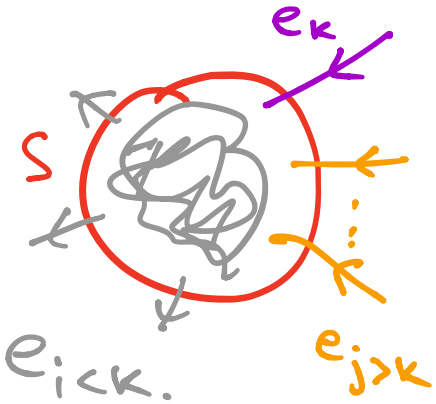
- F had no other edges in $\delta^-(S)$ prior to adding e_k

  (by construction)

  $\Rightarrow$ all edges of F in $A \cap \delta^-(S)$ are e_j for $j > k$.

- When S was chosen, F strongly connected within S.

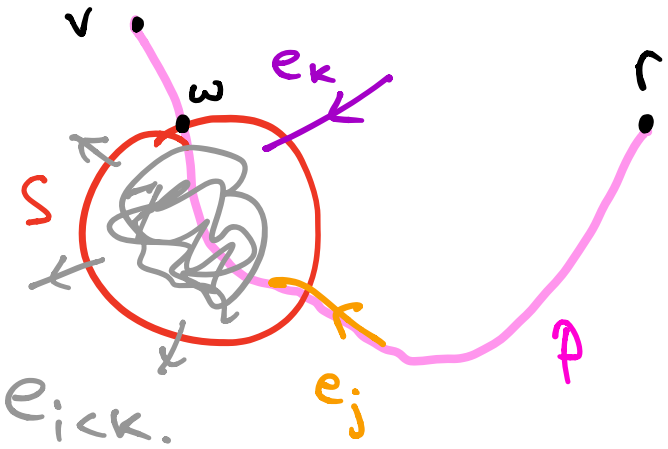
  $\Rightarrow$ S strongly connected using only e_i, $i < k$.



- **Subclaim**: e_j should have been removed in phase 2.

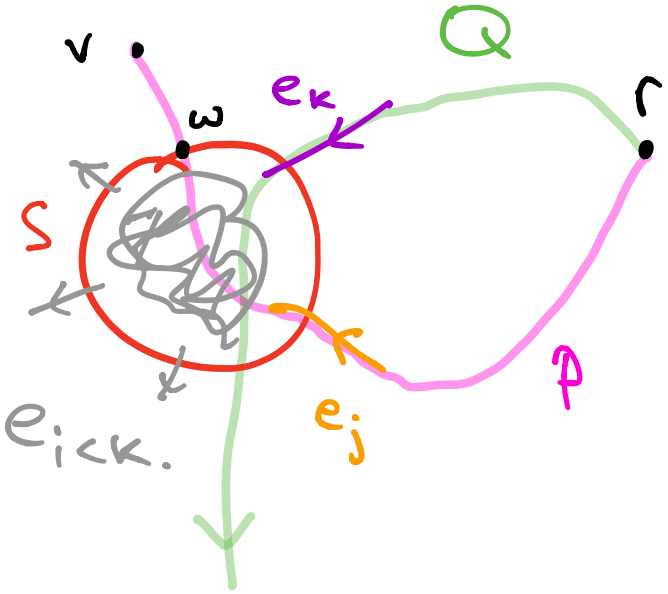
Why? Suppose a_j needed to visit v;

- Let P path $r \to v$ through e_j.
- Let w last vertex in S on P.



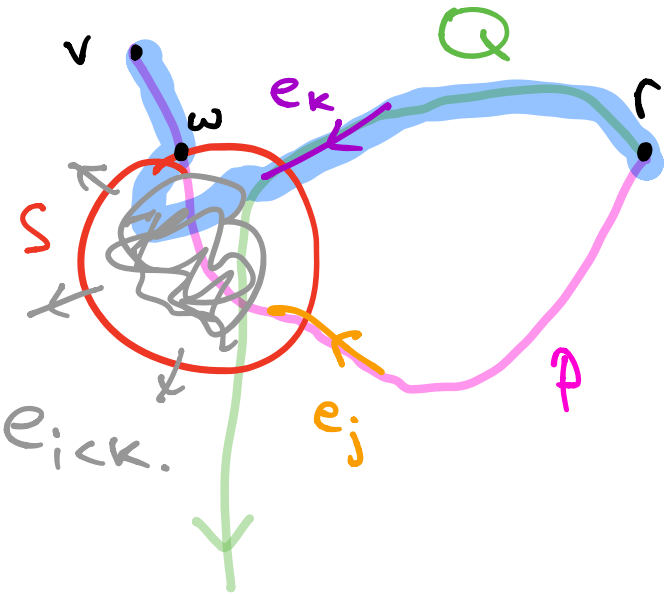
Note: P first enters S through e_j; else e_j could be shortcutted by strong connectivity of S at step j of phase 2.

- Because e_k necessary at step k of P_2,
  $\exists$ another path Q through
  $e_{k\,j}$ (which is also there at step j).



similarly: Q enters S first through e_k.

- Can use Q to shortcut P_j;
  thus e_j not necessary.


v
w
e_k
Q
r
S
$e_{i<k}$.
e_j
P

□.